Aaden Gabriel Abarintos

Susan Melichar

COP 2373

21 September 2025

Programming Exercise 2

**Programming Description:**

This program writes an application that creates a list of 30 keywords/phrases commonly found in spam, also known as phishing or junk mail, and receives the user's email as input. The program then looks at each individual word in the email to check if it is in the list. After doing so, the program then displays the likelihood that the email is spam, the “score” of the email—basically how many keywords are in the email—and counts how many phrases are in the email.

**Global Variables:**

1. None.

**Functions:**

1. *Function Name*: scan\_message()
   1. Description: This function scans the email for potential spam keywords, and counts how many keywords appear and keeps track of the spam score.
   2. Parameters:
      1. message: the email text that the user will input to check for spam.
      2. keywords: the list of 30 spam keywords/phrases that the program will check for in the message.
   3. Variables:
      1. msg\_lower: the input of the user, except all the letters are converted into lowercase.
      2. words: the list of the original words from the message as a list of individual terms.
      3. matches: an empty dictionary that stores how many spam words there are.
      4. score: accumulator variable to keep count of the “spam score.”
   4. Logical Steps:
      1. Initialize the variables
      2. Create a for loop to look through each keyword in the list and check if it is within the email.
      3. Create an if statement to make sure that a keyword that is not a singular keyword but is a phrase is counted.
      4. Create another if statement to store matched keywords in the dictionary and add points to the “spam score.”
      5. Return the spam score and the number of matched keywords.
   5. Returns: The function returns the score and the number of matched keywords.
2. *Function Name*: rate\_score()
   1. Description: This function uses the returned score from scan\_message() and uses that score to give an appropriate spam likelihood message.
   2. Parameters:
      1. Score: the value returned from scan\_message() representing how many spam keywords/phrases were found.
   3. Variables:
      1. score: uses the return from scan\_message() in an if statement to give an appropriate spam risk message.
   4. Logical Steps:
      1. Create an if statement to check which range the score variable falls into.
      2. Return the appropriate message.
   5. Returns: Returns a message depending on the spam score that the email initially received in the scan\_message() function.
3. *Function Name:* main()
   1. Description: This function creates the list of keywords and then asks the user to input their email they want to check for spam. Then, the function calls the scan\_message() and rate\_score() functions, respectively. Then the main function displays the return data from the other functions called.
   2. Parameters: None
   3. Variables:
      1. spam\_keywords: the list of 30 keywords and phrases common in spam mail.
      2. message: the prompt where the user is asked to input their email.
      3. score: uses the return of the score from the scan\_message() function and uses it within the main to display data.
      4. matches: uses the return of the scan\_message() function and uses it in order to make sure there are even matched keywords and phrases to display.
      5. category: uses the return of rate\_score() in order to display the correct spam likelihood message.
   4. Logical Steps:
      1. Initialize the list.
      2. Ask the user to paste their email.
      3. Create an if statement to make sure the input is not blank.
      4. Display the data.
   5. Returns: None

**Logical Steps:**

1. Define the functions
2. Call the main function
3. The scan\_message() function is called within the main function.
4. The rate\_score() function is called within the main function.

**Repository Link:** <https://github.com/7aaden/COP2373-Semester-1>

**Output Screenshot:**

